Sql\_class\_query

-- DDL - create , alter , truncate , drop

**create database ds\_class;**

**use ds\_class;**

create table dsStudents1 (

studentid int,

student\_name char(50),

student\_email varchar(100),

joining\_date date,

short\_desc text,

marks decimal(3,2) -- 000.00

);

select \* from dsstudents1;

**describe dsstudents1;**

**drop table dsStudents;**

**drop database ds\_class;**

**alter table dsstudents1 add weight float;**

alter table dsstudents drop column short\_desc;

alter table dsstudents modify weight int;

alter table dsstudents rename column weight to mass;

describe dsstudents;

select \* from dsstudents;

insert into dsstudents value (1,"deepak suneja", "deepak123@gmail.com", "2023-11-11", 8.4 ,67);

select \* from dsstudents;

truncate dsstudents;

select \* from insurance\_data;

truncate insurance\_data;

drop table insurance\_data;

use ds\_class;

-- create table tablename (

-- columnname1 datatype constraint,

-- columnname2 datatype constraint,

-- columnname3 datatype constraint

-- )

drop table dsStudents1;

create table teachers(

t\_id int,

t\_name char(50),

primary key (t\_id)

);

use ds\_class;

create table dsStudents1 (

studentid int ,

student\_name char(50) not null,

student\_email varchar(100) unique,

joining\_date date default('2023-11-05'),

short\_desc text,

marks decimal(3,2), -- 0.00

teacher\_id int,

primary key (studentid),

foreign key (teacher\_id) references teachers(t\_id)

);

alter table tablename add foreign key (teacher\_id) references teachers(t\_id);

describe dsStudents1;

-- DML - insert , update , delete

**-- insert into tablename(column names) value ();**

insert into teachers value (6, 'sudhanshu');

insert into teachers (t\_id) value (3);

insert into teachers values (4, "Ekta"),(5, 'Krish');

select \* from teachers;

**-- update my teachers table and set t\_name value as nitin where t\_id is equal to 2**

**update teachers set t\_name = "nitin" where t\_id = 2;**

**-- delete values from the teachers table whereever you find t\_id as 3**

**delete from teachers where t\_id = 3 ;**

**-- DQL - select important ds, da**

show databases;

show tables;

select \* from teachers;

select t\_id from teachers;

**-- operations - from , join , where , group by , having , select , order by , limit**

use ds\_class;

**-- operators -**

use mavenmovies;

**-- Arithmatic operator -- + - \* / %**

select \* from payment;

select amount, amount \* amount as doubled\_amount from payment;

select amount from payment where amount = 2\*amount/2; -- not work

**-- comparsion operator - < > = <> <= >=**

select \* from film;

select \* from film where replacement\_cost >= 20;

select \* from film where rating <> 'PG'; – < > → refer as not equal

**-- logical operator -- and , or , not , in , between**

select \* from film where **not** rating != 'PG';

select \* from film where length **between** 80 and 100;

select \* from film where length **in** (80 , 90 , 100);

select \* from film where length=80 **or**  length=90 or length= 100;

**-- constraints - set of rules**

**-- not null , default , unique, check, primary key , foregin key , candidate key , super key ,**

**-- composite key , artifical key**

**Details keys:** [**https://www.geeksforgeeks.org/types-of-keys-in-relational-model-candidate-super-primary-alternate-and-foreign/**](https://www.geeksforgeeks.org/types-of-keys-in-relational-model-candidate-super-primary-alternate-and-foreign/)

describe film;

create table dsStudents1 (

**-- addharcard + last\_nam of student**

student\_name char(50) **not nul**l,

adhaar\_card int , **-- candidate key**

student\_email varchar(100) **unique**,

joining\_date date **default**('2023-11-05'),

age int **check**(age>=18),

short\_desc text,

marks decimal(10,2), -- 00000000.00

teacher\_id int,

**primary key** (studentid),

**foreign key** (teacher\_id) references teachers(t\_id)

);

**-- Retrieve the titles of films that have a rental rate greater than $4.00.**

**select title , rental\_rate from film where rental\_rate > 4;**

**-- List the customer names and emails of those who live in the address with the ID 300.**

**select first\_name , last\_name , email , address\_id from customer where address\_id = 300;**

**-- Find the films with a length between 120 and 150 minutes (inclusive). Display the title and length.**

**select title , length from film where length between 120 and 150;**

**-- wildcard operator - like - \_ % - text data**

**-- Retrieve the first and last names of customers whose last name contains with the letter 'S'** .

select first\_name , last\_name from customer where last\_name like "S%";

select first\_name , last\_name from customer where last\_name like "%S";

select first\_name , last\_name from customer where last\_name like "%S%";

select first\_name , last\_name from customer where last\_name like "S%S";

select first\_name , last\_name from customer where last\_name like "%es%";

**-- Retrieve the first and lastname of customer whose first name should end with e and start with a?**

select first\_name , last\_name from customer where first\_name like "a%e";

**-- Retrieve the first and lastname of customer whose first name where letter a is on second position?**

select first\_name , last\_name from customer where first\_name like "\_a%";

**-- Retrieve the first and lastname of customer whose first name have letter a is on second position and**

**-- e on 4th position?**

select first\_name , last\_name from customer where first\_name like "\_a\_e%";

**-- Retrieve the first and lastname of customer whose first name have letter a is on second position and last name have letter e on 4th position?**

select first\_name , last\_name from customer where first\_name like "\_a%" and last\_name like "\_\_\_e%";

select \* from address;

select concat(district, address) from address;

**-- SQL pre defined functions mathematical/aggregate functions, strings, datetime**

select sum(amount) from payment; **-- summing all the column values and giving single** output

select avg(amount) from payment; **-- averaging all the column values and giving single** output

select count(\*) from payment; **-- counting all the values in column**

select min(amount) from payment; **-- minimum**

select max(amount) from payment; **-- maximum**

select concat(first\_name, " " , last\_name) as full\_name from actor;**-- concat to join multiple strings together**

select concat(first\_name, " " , last\_name) as full\_name,

length(concat(first\_name, last\_name)) as length from actor; **-- length to find number of char in text**

select lower(first\_name) from actor; -- upper , **lower to convert text into upper or lower case letters**

select upper("SaHiL");

select substr(address, 3, 10) from address; **-- substr( text data, starting point, no. of char)**

**-- its a function to extract/slice text data**

select substr("Sahil Garg", 3,5);

**-- datetime functions - month, year, day , monthname , datediff**

select rental\_date, day(rental\_date) from rental;

select return\_date, rental\_date, datediff(return\_date, rental\_date) from rental; **-- datediff is used to find difference between dates**

**-- What smallest rental duration ?**

select min(rental\_duration) from film;

**-- What is the highest replacement cost amongst all the films**

select max(replacement\_cost) from film;

**-- Display all films whose title length is greater than 10 characters**

select title, length(title) from film where length(title) > 10;

**-- Provide the count of unique ratings of films provided**

select count(distinct(rating)) from film;

**-- Display the list of first 4 cities which start and end with ‘a’**

select \* from city where city like "a%a" limit 4;

**-- operations - from , join , where , group by , having , select , order by , limit**

**-- Group by**

select \* from film;

select rating , sum(rental\_duration) from film group by rating;

select rating, avg(rental\_duration) from film where rating = 'PG';

select actor\_id , count(film\_id) as num\_of\_films from film\_actor group by actor\_id order by num\_of\_films desc;

**-- List the total sales amount for each customer in the database**

select \* from payment;

SELECT

customer\_id, SUM(amount) AS total\_amount

FROM

payment

WHERE

staff\_id = 2

GROUP BY customer\_id

HAVING total\_amount > 100;

**-- Joins - inner join , left join , right join and full join**

**-- T1 t2**

**-- stid name age stid tid subject**

**-- 1 abc 12 2 3 maths**

**-- 2 bdc 23 4 3 maths**

**-- 3 ojn 25 5 4 science**

**-- 4 ljn 26**

**-- right join - stid**

**-- stid name age stid tid subject**

**-- 2 bdc 23 2 3 maths**

**-- 4 ljn 26 4 3 maths**

**-- nul nul nul 5 4 science**

**--**

select \* from actor;

select \* from actor\_award;

select \* from actor inner join actor\_award on actor.actor\_id = actor\_award.actor\_id;

**-- Display all the cities with there postal code in India ?**

select \* from country; **-- country data**

select \* from city; **-- city data**

select \* from address; **-- postal codes**

SELECT

postal\_code, city, country

FROM

address

INNER JOIN

city ON address.city\_id = city.city\_id

INNER JOIN

country ON city.country\_id = country.country\_id

WHERE

country = 'India';

**-- Display the names of actors and the names of the films they have acted in.**

select \* from actor; **-- names of actor**

select \* from film\_actor; **-- common connection**

select \* from film; **-- title**

SELECT

concat(a.first\_name, " ", a.last\_name) as Name, title

FROM

actor a

INNER JOIN

film\_actor ON a.actor\_id = film\_actor.actor\_id

INNER JOIN

film ON film.film\_id = film\_actor.film\_id;

**-- Display all the actors and total number of films they have acted in.**

SELECT

concat(first\_name, " ", last\_name) as Name, count(title) as count\_of\_movies

FROM

actor

left JOIN

film\_actor ON actor.actor\_id = film\_actor.actor\_id

left JOIN

film ON film.film\_id = film\_actor.film\_id

group by Name

order by count\_of\_movies;

**-- Subquery -- non-correlated subqueries**

select avg(amount) from payment;

select payment\_id , amount from payment where amount > (select avg(amount) from payment) ;

select \* from film;

select \* from language;

select film\_id , title from film where language\_id in (select language\_id from language where name like "%e%");

select film\_id , title , language.language\_id from film inner join language on film.language\_id = language.language\_id where name like "%e%";

**-- Find the films that have a higher rental rate than any film in the 'Action' genre.**

**-- Display the title and rental rate.**

select \* from film; -- title , rental\_rate

select name from category;

select film\_id from film\_category where category\_id = (select category\_id from category where name = 'Action');

select max(rental\_rate) from film where film\_id in (select film\_id from film\_category

where category\_id = (select category\_id from category where name = 'comedy'));

select title, rental\_rate from film where rental\_rate > (

select max(rental\_rate) from film where film\_id in (select film\_id from film\_category

where category\_id = (select category\_id from category where name = 'Action')));

**-- Retrieve the films that have a replacement cost higher than the average replacement cost**

**-- of films in the 'Drama' genre. Display the title and replacement cost.**

select \* from film; -- title and replacement cost

select category\_id from category where name = 'Drama';

select film\_id from film\_category where category\_id = (select category\_id from category where name = 'Drama');

select avg(replacement\_cost) from film where film\_id in

(select film\_id from film\_category where category\_id = (select category\_id from category where name = 'Drama'));

select title , replacement\_cost from film where replacement\_cost>(

select avg(replacement\_cost) from film where film\_id in (select film\_id from film\_category

where category\_id =(select category\_id from category where name ='Drama')));

**-- Retrieve the titles of films that have a rental duration greater than the average rental duration of all films.**

select avg(rental\_duration) from film;

select title from film where rental\_duration > (select avg(rental\_duration) from film) order by rental\_duration desc;

**-- List the names of actors who have not appeared in any films. Display the actor's first name and last name.**

select first\_name , last\_name from actor where actor\_id not in (select actor\_id from film\_actor);

**-- List the first and last names of customers who have made payments.**

select first\_name , last\_name from customer where customer\_id in ( select distinct customer\_id from payment);

**-- Find the titles of films that are available in the inventory.**

select title from film where film\_id in (select distinct film\_id from inventory);

select first\_name , last\_name from actor where actor\_id in (select actor\_id from film\_actor

where film\_id = ( select film\_id from film where title = "ACADEMY DINOSAUR")); **-- movie name is not availale**

describe actor;

**-- correlated related queries**

**-- Retrieve the titles of films that have a rental rate higher than the average rental rate of films released in the same year.**

select title from film f where rental\_rate > (select avg(rental\_rate) from film where release\_year = f.release\_year );

**-- List customers who have made more than five payments. Display their first name and last name.**

select customer\_id , count(\*) from payment group by customer\_id having count(\*) > 5;

SELECT

first\_name, last\_name

FROM

customer

WHERE

(SELECT

COUNT(\*)

FROM

payment

WHERE

payment.customer\_id = customer.customer\_id) > 5;

SELECT

first\_name, last\_name, COUNT(\*)

FROM

customer

JOIN

payment ON customer.customer\_id = payment.customer\_id

GROUP BY first\_name , last\_name

HAVING COUNT(\*) > 5;

**-- Find films with a rental rate higher than the average rental rate for all films. Display the title and rental rate.**

select title, rental\_rate from film where rental\_rate > (select avg(rental\_rate) from film);

**-- Retrieve the names of actors who have appeared in at least two films. Display their first name and last name.**

select actor\_id , count(film\_id) from film\_actor group by actor\_id having count(film\_id) > 2;

select first\_name , last\_name from actor a where (select count(film\_id) from film\_actor fa where fa.actor\_id = a.actor\_id) >= 2;

**-- Write a query to count the number of film rentals for each customer and**

**-- the containing query then retrieves those customers name who have rented exactly 30 films.**

select first\_name , last\_name from customer where (select count(rental\_id) from rental where rental.customer\_id = customer.customer\_id) = 30;

**-- Write a query to find all customers whose total payments for all film rentals are between 100 and 150 dollars.**

select first\_name , last\_name from customer c where (select sum(amount) from payment p where c.customer\_id = p.customer\_id) between 100 and 150;

select customer\_id , count(customer\_id) as tot\_rentals , sum(amount) as tot\_payments

from payment group by customer\_id;

**-- Write a query to generate a list of customer IDs along with the number of film rentals and the total payments.**

select c.first\_name, c.last\_name , tot\_rentals , tot\_payments from customer c inner join

( select customer\_id , count(customer\_id) as tot\_rentals , sum(amount) as tot\_payments

from payment group by customer\_id) a

on c.customer\_id = a.customer\_id;

use mavenmovies;

**-- inner join**

select count(\*) from film;

select count(\*) from film\_actor;

select count(\*) from film as f inner join film\_actor as fa on f.film\_id = fa.film\_id inner join actor as a on

a.actor\_id = fa.actor\_id;

**-- left join**

select \* from film as f left join film\_actor as fa on f.film\_id = fa.film\_id ;

**-- right join**

select count(\*) from film as f right join film\_actor as fa on f.film\_id = fa.film\_id;

**-- cross join**

select \* from film cross join film\_actor;

**-- full outer join**

-- select count(\*) from film as f full outer join film\_actor as fa on f.film\_id = fa.film\_id;

select \* from film as f left join film\_actor as fa on f.film\_id = fa.film\_id union

select \* from film as f right join film\_actor as fa on f.film\_id = fa.film\_id;

**-- natural join**

select \* from film natural join film\_category;

**-- non - correlated query**

select \* from payment;

**-- give me the customer id who is spending more than average**

select avg(amount) from payment;

select distinct customer\_id from payment where amount > 4.2;

select distinct customer\_id from payment where amount > (select avg(amount) from payment);

**-- retieve the film title , desc, and release year for the film that has the longest duration**

select \* from film;

select title , description , release\_year from film where length = (select max(length) from film);

**-- scaler subquery**

**-- show the title , rental\_rate and diff from average renatl rate for each film**

select title , rental\_rate , rental\_rate - (select avg(rental\_rate) from film) as average\_rental\_diff from film;

**-- multirow subquery**

**-- Write a query to generate a list of customer IDs along with the number of film rentals and the total payments.**

select c.first\_name, c.last\_name , tot\_rentals , tot\_payments from customer c inner join

( select customer\_id , count(customer\_id) as tot\_rentals , sum(amount) as tot\_payments

from payment group by customer\_id) a

on c.customer\_id = a.customer\_id;

( select customer\_id , count(customer\_id) as tot\_rentals , sum(amount) as tot\_payments

from payment group by customer\_id);

**-- Determine the customers who have rented movies from both stores (store ID 1 and store ID 2)**

select \* from customer; -- name

select \* from inventory; -- store\_id

select \* from rental; -- customer\_id

select \* from customer as c inner join rental as r on c.customer\_id = r.customer\_id

inner join inventory as i on r.inventory\_id = i.inventory\_id ;

SELECT

c.customer\_id, c.first\_name, c.last\_name

FROM

customer c

INNER JOIN

rental r ON c.customer\_id = r.customer\_id

INNER JOIN

inventory i ON r.inventory\_id = i.inventory\_id

INNER JOIN

store s ON i.store\_id = s.store\_id

WHERE

s.store\_id IN (1 , 2)

GROUP BY c.customer\_id , c.first\_name , c.last\_name

HAVING COUNT(DISTINCT s.store\_id) = 2;

**-- Create a CTE with two named subqueries. The first one gets the actors with last names starting with s.**

**-- The second one gets all the pg films acted by them. Finally show the film id and title.**

**-- subquery 1**

select \* from actor where last\_name like "s%" ;-- saving the result in variable - actor\_s

**-- sub query 2**

select a.actor\_id , a.first\_name , a.last\_name , f.title , f.film\_id from actor as a inner join film\_actor as fa on a.actor\_id = fa.actor\_id

inner join film as f on f.film\_id = fa.film\_id where rating = 'pg'; -- actor\_pg

with actor\_s as (

select \* from actor where last\_name like "s%"),

actor\_pg as (

select a.actor\_id , a.first\_name , a.last\_name , f.title , f.film\_id from actor\_s as a inner join film\_actor as fa on a.actor\_id = fa.actor\_id

inner join film as f on f.film\_id = fa.film\_id where rating = 'pg')

select film\_id , title from actor\_pg;

**-- Add one more subquery to the previous CTE to get the revenues of those movies**

with actor\_s as (

select \* from actor where last\_name like "s%"), -- task 1 - filter people with last name as 's'

actor\_pg as (

select a.actor\_id , a.first\_name , a.last\_name , f.title , f.film\_id from actor\_s as a inner join film\_actor as fa on a.actor\_id = fa.actor\_id

inner join film as f on f.film\_id = fa.film\_id where rating = 'pg'), -- task 2 - filter film with rating as 'pg'

film\_revenue as (

select ap.film\_id , ap.title , p.amount from actor\_pg as ap inner join inventory as i on ap.film\_id = i.film\_id inner join

rental as r on r.inventory\_id = i.inventory\_id inner join payment as p on p.rental\_id = r.rental\_id

) **-- task 3 - join amount of filter films**

select title , sum(amount) from film\_revenue group by title; **-- task 4 - total reveniue of each movie**

**-- Question 1 - Give me the details of films along with its revenue having pg rating and actor with last name as s ?**

**-- number of tables - actor , film\_actor , film, inventory , rental , payment**

**-- Write a query to generate a value for the activity\_type column which returns the string “Active” or “Inactive”**

**-- depending on the value of the customer.active column**

select \* from customer;

select first\_name , last\_name ,

case

when active = 1 then 'active'

else 'inactive'

end as activity\_type

from customer;

**-- Write a query to show the number of film rentals for May, June and July of 2005 in a single row.**

select monthname(rental\_date)as month\_name , count(rental\_id) from rental group by month\_name;

select \* from rental;

select sum(case when monthname(rental\_date) = 'May' then 1 else 0 end) as may,

sum(case when monthname(rental\_date) = 'June' then 1 else 0 end) as june ,

sum(case when monthname(rental\_date) = 'July' then 1 else 0 end) as July from rental;

**-- Write a query to categorize films based on the inventory level.**

**-- If the count of copies is 0 then ‘Out of stock’**

**-- If the count of copies is 1 or 2 then ‘Scarce’**

**-- If the count of copies is 3 or 4 then ‘Available’**

**-- If the count of copies is >= 5 then ‘Common’**

select title ,

case

(select count(\*) from inventory where inventory.film\_id = film.film\_id )

when 0 then 'out of stock'

when 1 then 'Scares'

when 2 then 'Scares'

when 3 then 'Available'

when 4 then 'Available'

else 'common' end film\_availabiliy

from film;

**-- Write a query to create a single row containing the number of films based on the ratings** (G, PG and NC17)

SELECT

SUM(CASE WHEN rating = 'G' THEN 1 ELSE 0 END) AS G\_count,

SUM(CASE WHEN rating = 'PG' THEN 1 ELSE 0 END) AS PG\_count,

SUM(CASE WHEN rating = 'NC-17' THEN 1 ELSE 0 END) AS NC17\_count

FROM film;

select distinct rating from film ;

--

select customer\_id from customer where customer\_id in (select customer\_id from rental group by customer\_id

having count(rental\_id) > ( select avg(num\_rentals) from (select customer\_id , count( rental\_id) as num\_rentals from rental group by

customer\_id ) as rental\_counts

));

with rental\_count as (

select customer\_id , count( rental\_id) as num\_rentals from rental group by customer\_id),

avg\_value as (

select avg(num\_rentals) from rental\_count),

filter\_custemer as

( select \* from rental\_count where num\_rentals > (select \* from avg\_value))

select \* from filter\_custemer;

describe film\_actor;

select count(\*) from film\_actor;

**-- Write a query to count the number of film rentals for each customer**

**-- and the containing query then retrieves those customers who have rented exactly 30 films.**

**-- customer , rental**

select \* from customer;

select \* from rental;

select first\_name , last\_name from customer where (select count(\*) from rental where rental.customer\_id = customer.customer\_id) = 30;

**-- Write a query to find all customers whose total payments for all film rentals are between 100 and 150 dollars.**

**-- customer , payment**

**-- retieve the film title along with their description and lengths that have rental rate greater than avg rental\_rate f films**

**-- released in same year**

select title , description ,length from film as f where rental\_rate > (

select avg(rental\_rate) from film where release\_year = f.release\_year);

select first\_name , last\_anme from customer where customer\_id in ( select customer\_id from rental group by customer\_id having count(rental\_id)

> (select avg(rental\_count) from (select count(rental\_id) as rental\_count from rental group by customer\_id) as avg\_rentals));

select customer\_id , count(rental\_id) as film\_count from rental group by customer\_id having count(rental\_id) >= 5;

with agr\_film\_actor as (

select title , count(fa.actor\_id) as count\_of\_actor from film\_actor fa inner join film f on f.film\_id = fa.film\_id group by f.title

)

select title , count\_of\_actor ,

case

when count\_of\_actor > 5 then "big star cas" else "small start cast" end as star\_cast from agr\_film\_actor;

with recursive even\_no (n) as (

select 10 **-- anchor**

union all

select n+2 from even\_no where n < 20 **-- recursive**

)

select \* from even\_no;

with recursive odd\_no (n) as (

select 6 **-- anchor**

union all

select 2\*n+1 from odd\_no where n < 25 **-- recursive**

)

select \* from odd\_no;

CREATE TABLE emp (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(50),

manager\_id INT,

FOREIGN KEY (manager\_id) REFERENCES emp(emp\_id)

);

INSERT INTO emp (emp\_id, emp\_name, manager\_id) VALUES

(1, 'John Doe', NULL), **-- Top-level employee**

(2, 'Alice Smith', 1), **-- Employee managed by John Doe**

(3, 'Bob Johnson', 1), **-- Employee managed by John Doe**

(4, 'Charlie Brown', 2), **-- Employee managed by Alice Smith**

(5, 'Diana Williams', 2), **-- Employee managed by Alice Smith**

(6, 'Eva Garcia', 3), **-- Employee managed by Bob Johnson**

(7, 'Franklin Wang', 3); **-- Employee managed by Bob Johnson**

with recursive emp\_hirarchy as (

select emp\_id , emp\_name , manager\_id , 0 as level from emp where manager\_id is null

union all

select e.emp\_id , e.emp\_name , e.manager\_id , eh.level+1 from emp as e join emp\_hirarchy as eh on e.manager\_id = eh.emp\_id)

select emp\_id , emp\_name , manager\_id , level from emp\_hirarchy;

**-- calculate the average rental duration for films and display both the individual average duration**

**-- for each film and the overall average duration.**

select \* from film;

select film\_id , rating, title , rental\_duration , avg(rental\_duration) over() ,

avg(rental\_duration) over (partition by rating)

from film;

**-- assign the ranks based on ratings alog with rental\_durations**

select rating, rental\_duration ,

rank() over (partition by rating order by rental\_duration) as rankings

from film;

**-- Retrieve the top 5 films with the highest rental rates, displaying the film title, rental rate, and the average rental**

**-- rate for all films. Use window functions to rank the films based on their rental rates.**

select title, rental\_rate,

avg(rental\_rate) over(),

rank() over(order by rental\_rate asc)

from film

order by rental\_rate desc

limit 5;

**-- Find the total number of rentals for each customer, along with their -- 3:27 - 3:32**

**-- individual rental counts and the average number of rentals across all customers. -- rental == customer\_id , count(rental) , avg(count())**

**-- add new column with individual having higher value than avg is marked higher or else lower**

**-- give me the list of people who rented movies higher than average -- 3:36 - 3:40**

select customer\_id , count(rental\_id) as total\_count, avg(count(rental\_id)) over() as avg\_count,

case when count(rental\_id) - avg(count(rental\_id)) over() > 0 then "higher" else "lower" end count\_stats

from rental group by customer\_id order by count\_stats;

**-- Find the films that have been rented the least number of times. Display the film**

**-- title, rental count, and the average rental count across all films.**

**-- from - join - where - group by - having - select - order by - limit**

select title , count(rental\_id) as no\_of\_rental, avg(count(rental\_id)) over() from

film f left join inventory i on f.film\_id = i.film\_id

left join rental r on r.inventory\_id = i.inventory\_id group by title order by no\_of\_rental;

**-- Display the title and rental duration of films from the database,**

**-- alongside the title and rental duration of the subsequent film (using LEAD function)**

select title , rental\_duration , lag(title, 7) over(order by rental\_duration),

lag(rental\_duration, 2) over(order by rental\_duration) from film;

**-- lag -- when you want to compare performance on weekly bases**

**-- function - string , numeric , date function , aggregate , window functions**

select concat(first\_name , " " , last\_name) from actor;

**-- user defined functions**

**-- create a function to get a squared value**

**-- SET GLOBAL log\_bin\_trust\_function\_creators = 1;**

**-- delimiter //**

**-- create function squared (num int)**

**-- returns int**

**-- begin**

**-- declare result int;**

**-- set result = num\*num;**

**-- return result;**

**-- end //**

**-- delimiter ;**

select squared(5);